ESP8266 AT Command Examples

Version 1.3

Espressif Systems IOT Team

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The WiFi Alliance Member Logo is a trademark of the WiFi Alliance.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

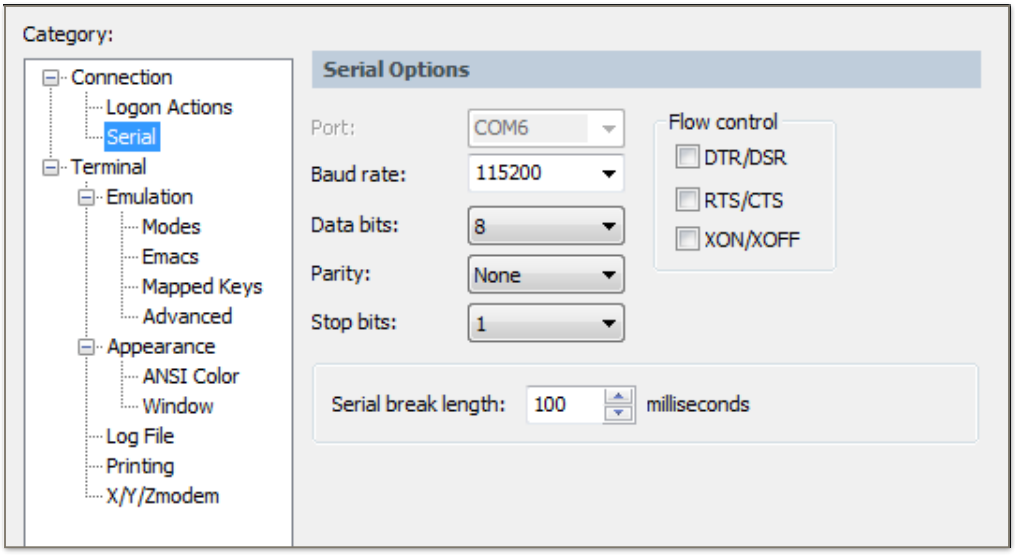
Table of Contents

1. Preambles

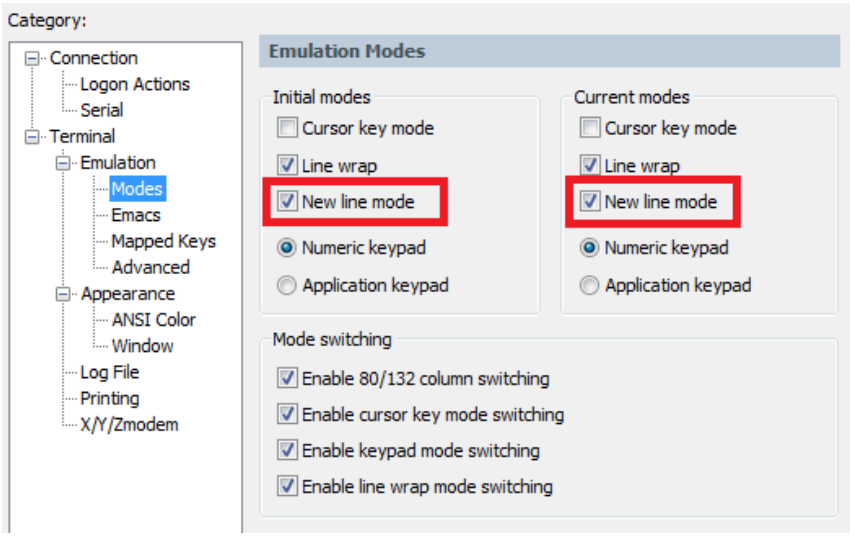
Herein we introduces some specific examples on the usage of Espressif AT Commands. For more information about the complete instruction set, please refer to documentation " 4A-ESP8266__AT Instruction Set ".

(1) Flash in AT bin files which supports AT commands (`/esp_iot_sdk/bin/at`) into the ESP8266 device, according to "readme.txt" there.

(2) Power on device and set serial baud rate to 115200. Enter AT commands.



Note: Please pay attention to the new line mode, AT command need "`/r/n`" to be the end.



2. Single Connection as TCP Client

- **Set WiFi mode**:

```
AT+CWMODE=3      // softAP+station mode
Response :OK
```

- **Connect to router**:

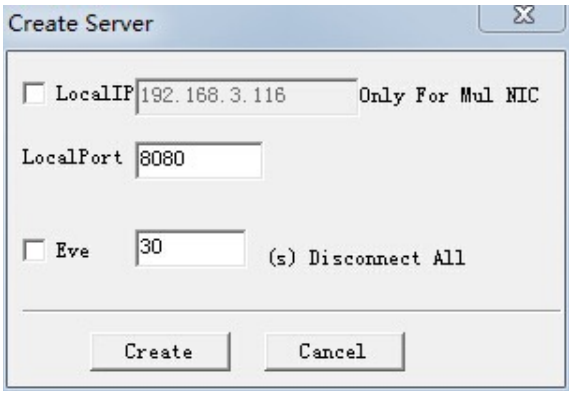
```
AT+CWJAP="SSID", "password"       // SSID and password of router
Response :OK
```

- **Query device's IP**:

```
AT+CIFSR
Response :192.168.3.106   // Device got an IP from router.
```

- Connect PC to the same router that ESP8266 is connected to.

  Using a network tool on the computer to create a server.



- **ESP8266 connect to server as a client**:

```
AT+CIPSTART="TCP", "192.168.3.116", 8080  //protocol、server IP & port
Response :OK
```

- **Send data**:

```
AT+CIPSEND=4   // set date length which will be sent,  such as 4 bytes

>DGFY         // enter the data,  no CR
Response :SEND OK
```

Note: If the number of bytes sent is bigger than the size defined (n), will reply busy, and after sending n number of bytes, reply SEND OK.

- **Receive data**:

```
    +IPD, n: xxxxxxxxxx    // received n bytes,  data=xxxxxxxxxx
```

3. UDP Transmission

UDP transmission is established via AT+CIPSTART. There is no such definition as UDP server or UDP client. For more information about AT commands, please refer to documentation " 4A-ESP8266__AT Instruction Set ".

- **Set WiFi mode** :

```
AT+CWMODE=3      // softAP+station mode
Response :OK
```

- **Connect to router**:

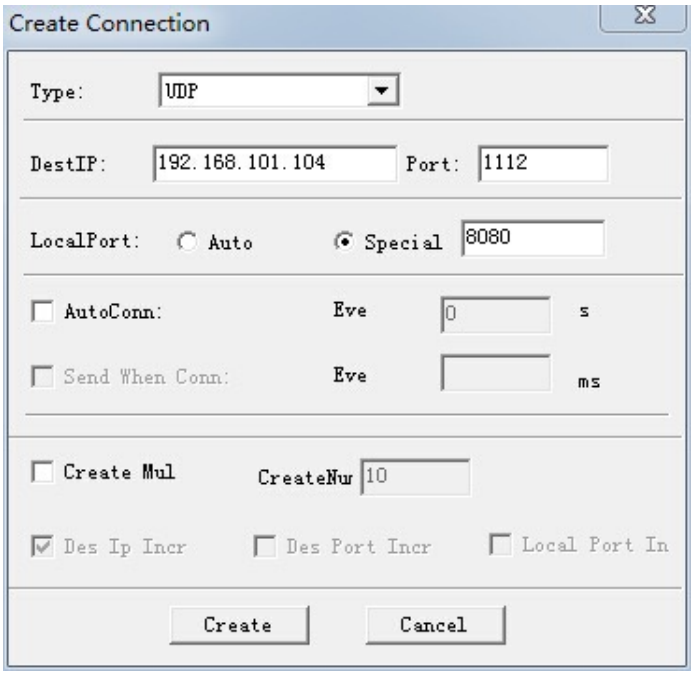
```
AT+CWJAP="SSID", "password"       // SSID and password of router
Response :OK
```

- **Query device's IP**:

```
AT+CIFSR
Response :+CIFSR: STAIP, "192.168.101.104" // IP address of ESP8266 station
```

- Connect PC to the same router as ESP8266 is connected to.

  Using a network tool on the computer to create a UDP .



Below is two examples on UDP transmission.

3.1. UDP (remote IP and port are fixed)

In UDP transmission, whether remote IP and port is fixed or not is decided by the last parameter of "AT+CIPSTART". "0" means that the remote IP and port is fixed and cannot be changed. A specific ID is given to such connection, making sure that the data sender and receiver will not be replaced by other devices.

- **Enable multiple connection**:

```
AT+CIPMUX=1
Response :OK
```

- **Create a UDP transmission, for example, ID is 4.**

```
AT+CIPSTART=4, "UDP", "192.168.101.110", 8080, 1112, 0
Response :4, CONNECT   OK
```

Note :

"192.168.101.110", 8080 here is the remote IP and port of UDP transmission of the opposite side, i.e., the configuration set by PC.

1112 is the local port of ESP8266. User can self-define this port. The value of this port will be random if it's not defined beforehand.

0 means that remote IP and port is fixed and cannot be changed. For example, if another PC also creates a UDP entity and sends data to ESP8266 port 1112. ESP8266 can receive data sent from UDP port 1112, but when data is sent using AT command "AT+CIPSEND=4, X", it will still be sent to the first PC end. If this parameter is not 0, it will send to a new PC.

- **Send data**:

```
AT+CIPSEND=4, 5    // Send 5 bytes to transmission NO.4
>DGFYQ          // enter the data,  no CR
Response :SEND OK
```

Note:

If the number of bytes sent is bigger than the size defined (n), will reply busy, and after sending n number of bytes, reply SEND OK.

- **Receive data**:

```
+IPD, 4, n: xxxxxxxxxx    // received n bytes,  data=xxxxxxxxxx
```

- **Delete UDP transmission NO.4**:

```
AT+CIPCLOSE=4
Response :4, CLOSED   OK
```

3.2. UDP (remote IP, port can be changed)

- **Create a UDP transmission, last parameter is "2".**

```
AT+CIPSTART="UDP", "192.168.101.110", 8080, 1112, 2
Response :CONNECT   OK
```

Note :

"192.168.101.110", 8080 here refer to the remote IP and port of UDP transmission terminal which is created on PC in step 4;

1112 is the local port of ESP8266. User can self-define this port. The value of this port will be random if it's not defined beforehand.

2 means the opposite terminal of UDP transmission side will change to be the latest one that has been communicating with ESP8266.

- **Send data**:

```
AT+CIPSEND=5   // Send 5 bytes

>DGFYQ         // enter the data,  no CR
Response :SEND OK
```

Note:

If the number of bytes sent is bigger than the size defined (n), will reply busy, and after sending n number of bytes, reply SEND OK.

- If you want to send data to any other UDP terminals, please set the IP and port of this terminal.

```
AT+CIPSEND=6, "192.168.101.111", 1000   // Send 6 bytes

>abcdef         // enter the data,  no CR
Response :SEND OK
```

- **Receive data**:

```
+IPD, n: xxxxxxxxxx    // received n bytes,  data=xxxxxxxxxx
```

- **Delete UDP transmission**:

```
AT+CIPCLOSE
Response :CLOSED   OK
```

4. Transparent Transmission

Transparent transmission is enabled only when ESP8266 is working as TCP client creating a single connection, or in UDP transmission.

4.1. TCP client single connection UART - WiFi passthrough

Here is an example that ESP8266 station as TCP client to create a single connection and execute transparent transmission. For ESP8266 soft-AP, it can execute transparent transmission in the similar way. For more information about AT commands, please refer to documentation " 4A-ESP8266__AT Instruction Set ".

- **Set WiFi mode** :

```
AT+CWMODE=3      // softAP+station mode
Response :OK
```

- **Connect to router**:

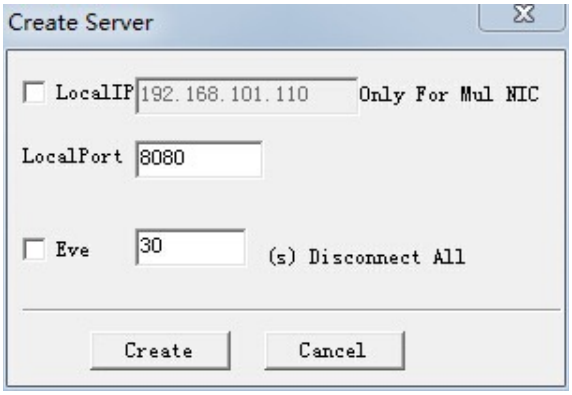
```
AT+CWJAP="SSID", "password"       // SSID and password of router
Response :OK
```

- **Query device's IP**:

```
AT+CIFSR
Response :192.168.101.105   // Device's IP that got from router.
```

- Connect PC to the same router that ESP8266 is connected to.

  Using a network tool on the computer to create a server.



- **Device connect to server**:

```
AT+CIPSTART="TCP", "192.168.101.110", 8080  // protocol、server IP & port
Response :OK     Linked
```

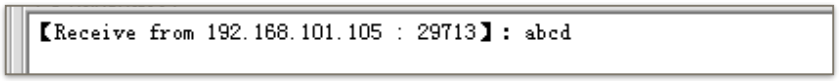
- **Enable transparent transmission mode**:

```
AT+CIPMODE=1
Response :OK
```

- **Start sending data**:

```
AT+CIPSEND
     Response:  >          //From now on,  data received from UART will be
transparent transmitted to server.
```

The network tool on PC will receive the data.



- **Stop sending data**:

If received a packet of data that contains only "+++", then the transparent transmission process will be stopped. Please wait at least 1 second before sending next AT command.

Please be noted that if you input "+++" directly by typing, the "+++", may not be recognised as three consecutive "+" because of the Prolonged time when typing.

Note:

The aim of ending "+++" is to exit transparent transmission and turn back to accept normal AT command, while TCP still remains connected. However, we can also use command "AT+CIPSEND" to turn back into transparent transmission.

- **Disable UART - WiFi passthrough mode (transparent transmission)**

```
AT+CIPMODE=0
Response :  OK
```

- **Delete TCP connection**:

```
AT+CIPCLOSE
Response :CLOSED   OK
```

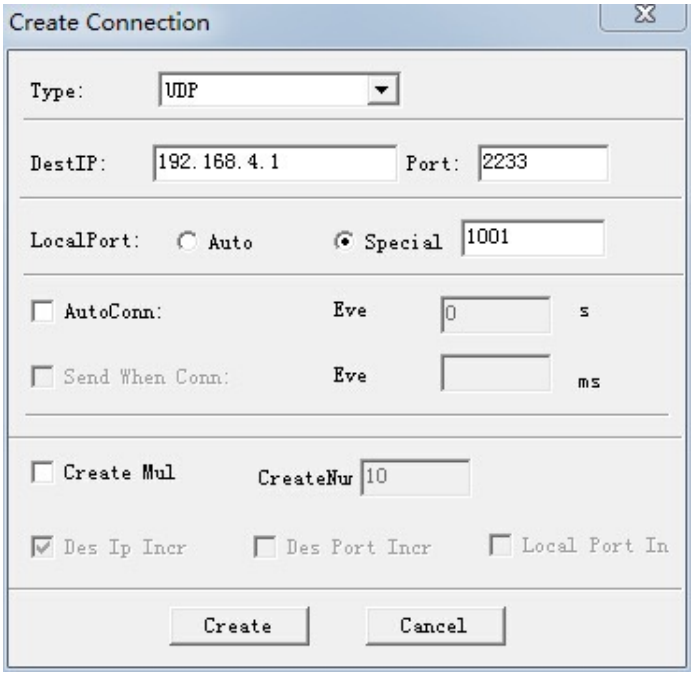
4.2. UDP transmission UART - WiFi passthrough

Here is an example that ESP8266 soft-AP create a UDP transparent transmission. For ESP8266 station, it can execute UDP transparent transmission in the similar way. For more information about AT commands, please refer to documentation " 4A-ESP8266__AT Instruction Set ".

- **Set WiFi mode** :

```
AT+CWMODE=3      // softAP+station mode
Response :OK
```

- **Connect PC to ESP8266 soft-AP**
- Using a network tool on PC to create a UDP.



- **Device create a UDP transmission which remote IP and port are fixed.**

```
AT+CIPSTART="UDP","192.168.4.2",1001,2233,0
Response :OK
```

- **Enable transparent transmission mode**:

```
AT+CIPMODE=1
Response :OK
```

- **Start sending data**:

```
AT+CIPSEND
```

```
        Response:  >          //From now on,  data received from UART will be
    transparent transmitted to server.
```

- **Stop sending data**:

If a packet of data that contains only "+++", then the transparent transmission process will be stopped. Please wait at least 1 second before sending next AT command.

Please be noted that if you input "+++" directly by typing, the "+++", may not be recognised as three consecutive "+" because of the Prolonged time when typing.

Note:

The aim of ending "+++" is to exit transparent transmission and turn back to accept normal AT command, while UDP transmission still exists. However, we can also use command "AT+CIPSEND" to turn back into transparent transmission.

- **Disable UART - WiFi passthrough mode (transparent transmission)**

```
    AT+CIPMODE=0
    Response :  OK
```

- **Delete UDP transmission**:

```
    AT+CIPCLOSE
    Response :CLOSED   OK
```

5. Multiple Connection as TCP Server

When ESP8266 is working as a TCP server, a multiple of connections shall be maintained. That is to say, there should be more than one client connecting to ESP8266.

Here is an example showing how TCP server is realized when ESP8266 is working in softAP mode:

- **Set WiFi mode** :

```
AT+CWMODE=3      // softAP+station mode
Response :OK
```

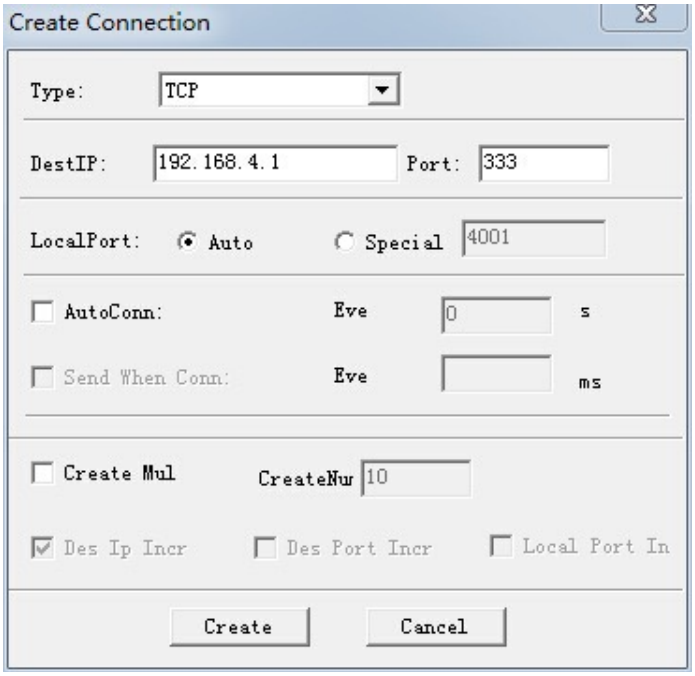
- **Enable multIPle connection**:

```
AT+CIPMUX=1
Response :OK
```

- **Setup server**:

```
AT+CIPSERVER=1       // default port = 333
Response :OK
```

- **Connect PC to ESP8266 soft-AP**
- Using a network tool on PC to create a TCP client and connect to ESP8266.



Note:

When ESP8266 is working as a TCP server, there exists a timeout mechanism. If the TCP client is connected to the ESP8266 TCP server, whereas there is no data transmission for a period of time, then the server will stop the connection with the client. To avoid such problems, please set up a data transmission circulation every 5 seconds.

- **Send data**:

```
// ID number of connection is defaulted to be 0.
 AT+CIPSEND=0, 4   // send 4 bytes to connection NO.0

 >iopd          // enter the data,  no CR
 Response :SEND OK
```

Note:

If the number of bytes sent is bigger than the size defined (n), will reply busy, and after sending n number of bytes, reply SEND OK.

- **Receive data**:

```
  +IPD, 0, n: xxxxxxxxxx    // received n bytes,  data = xxxxxxxxxx
```

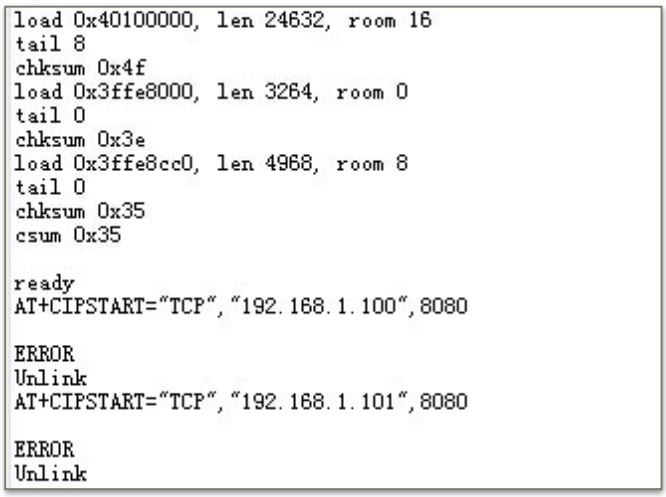
- **Delete one TCP connection**:

```
  AT+CIPCLOSE=0    // Delete NO.0 connection.
Response :0, CLOSED    OK
```

6. Questions & Answers

If you have any questions about the execution of AT instructions, please contact support-at@espressif.com. Please describe the issues that you encountered with information as follows:

- Version info or AT Command: You can use command "AT+GMR" to acquire your current AT command version info.
- Hardware Module info :example AITHINK ESP-01
- Screenshot of the test steps, for example:



- If possible, please provide the printed log information, such as:

```
ets Jan  8 2013, rst cause: 1,  boot mode: (3, 3)
load 0x40100000,  len 26336,  room 16
tail 0
chksum 0xde
load 0x3ffe8000,  len 5672,  room 8
tail 0
chksum 0x69
load 0x3ffe9630,  len 8348,  room 8
tail 4
chksum 0xcb
csum 0xcb
SDK version: 0.9.1
addr not ack when tx write cmd
```